Understanding Bootstrap Confidence Intervals

Yossi Levy

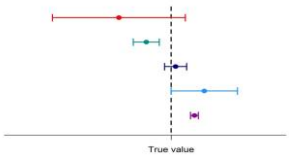
January 6, 2020

Outline

- What is a confidence interval
- Sampling basics
- The bootstrap principle
- Using the boot package
- Understanding the boot.ci output

What is a confidence interval

A confidence interval gives an estimated range of values which is **likely to include** an unknown population parameter, the estimated range being calculated from a given set of sample data.



Among all CIs with confidence level $1-\alpha$ the best CI is the shortest CI

Example – law schools data

- Data on 82 law schools (Efron and Tibshirani, 1993, page 21)
- For each school we have data on mean LSAT and mean GPA of the students
- TRUE correlation coefficient is 0.7599

```
> schools=read.csv(file="schools.csv")
> print(head(schools))
  school lsat  gpa
1      1  622 3.23
2      2  542 2.83
3      3  579 3.24
4      4  653 3.12
5      5  606 3.09
6      6  576 3.39
> true.r=cor(schools$lsat, schools$gpa)
> print(paste("Rho = True R =", substr(as.character(true.r),1,6)))
[1] "Rho = True R = 0.7599"
>
```


Example – law schools data

Sample of 15 out of 82 law schools (Efron and Tibshirani, 1993, page 19)

```
> print(school.sample)
   lsat  gpa
1   576 3.39
2   635 3.30
3   558 2.81
4   578 3.03
5   666 3.44
6   580 3.07
7   555 3.00
8   661 3.43
9   651 3.36
10  605 3.13
11  653 3.12
12  575 2.74
13  545 2.76
14  572 2.88
15  594 2.96
> # r is an esimate for the correlation coefficient in the population
> r=cor(school.sample$lsat, school.sample$gpa)
> print(r)
[1] 0.7763745
>
```


CI for the correlation coefficient

Under standard assumptions: $SE(r) = \sqrt{\frac{1-r^2}{n-2}} \sim t_{(n-2)}$

So CI for Rho is $r \pm t_{n-2,1-\alpha/1} \cdot SE(r)$

```
> n=nrow(school.sample)
> se.r=sqrt((1-r^2)/(n-2))
> t.value=qt(0.975, (n-2))
> ci.t=c(r-t.value*se.r,r+t.value*se.r)
> print(ci.t)
[1] 0.3987292 1.1540198
>
```

It is easy to check that the confidence lever of this CI is actually 99%.

But…

- The upper confidence limit is greater than 1
- The CI length is 0.755
- Even if we set the upper confidence limit at 1, the CI length is 0.601, which is not very good

Possible solution – Fisher's transformation

$$z = \log\frac{1+r}{1-r} \sim N\left(0,\ 1/\sqrt{n-3}\right)$$
$$r = \frac{e^{2z}-1}{e^{2z}+1}$$

```
> # Fisher's transformation
> fisher.z=0.5*log((1+r)/(1-r))
> se.fisher.z=1/sqrt(n-3)
> ci.fisher.z=c(fisher.z-1.96*se.fisher.z, fisher.z+1.96*se.fisher.z)
> # back transformation
> ci.fisher=(exp(2*ci.fisher.z)-1)/(exp(2*ci.fisher.z)+1)
> print(ci.fisher)
[1] 0.4385024 0.9219663
>
```

Fisher's CI length is 0.483

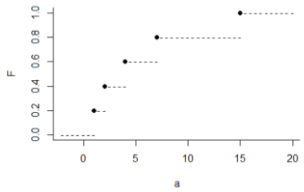
Sampling basics

- We are interested in estimating a parameter θ of a population distribution F
- We take a sample from the population: $X_1, X_2, \ldots, X_n$
- We assume that the samples are **Independent and Identically Distributed** random variables
- This can be achieved only if the sample was taken from the population **with replacement**
- From the sample we can estimate both F and θ
- We denote the estimators by $\hat{F}_n$ and $\hat{\theta}_n$

The empirical distribution function

- $\hat{F}_n$ is called the empirical distribution function of the sample
- $\hat{F}_n(a) = \frac{1}{n}\sum_{j=1}^{n} I(X_j \le a)$

```
> # empirical distribution function illustration
> sample=c(1,2,4,7,15)
> n=length(sample)
> x=0
> F=0
> for(j in 1:45){
+   a=0.5*j-2.5
+   x[j]=a
+   F[j]=length(which(sample<a))/n
+ }
> a=x
> plot(a, F, pch="-")
> points(sample, 0.2*1:5, pch=16)
>
```



The bootstrap principle

- Estimate $\hat{F}_n$ by taking a sample from the sample.
- Sampling should be with replacement, just as the original sample was assumed to be a sample with replacement
- Denote the estimate of $\hat{F}_n$ by F_n^*
- Since F_n^* is an estimate for $\hat{F}_n$, and $\hat{F}_n$ is an estimate for F, then F_n^* is an estimate for F
- Similarly, if we estimate θ^* from F_n^*, then θ^* is an estimate for $\hat{\theta}$, which is an estimate to θ
- If we take many samples from the sample, we can estimate the standard deviation of $\hat{\theta}$

Implementing the boot package

1. Define a function that calculate the statistic we want from a sample
2. Use the boot function to get R bootstrap replicates of the statistic
3. Use the boot.ci function to get the confidence intervals

Define statistic function

```
> # define a function that returns the statistic we want
> calc.r=function(data, indices, x, y) {
+   d=data[indices, ]
+   r=round(as.numeric(cor(d[x], d[y])),3)
+   return(r)
+ }
> r=calc.r(school.sample, 1:15, "lsat", "gpa")
> print(r)
[1] 0.776
> set.seed(123)
> boot.sample=sample(1:15, size=15, replace=TRUE)
> print(boot.sample)
 [1]  5 12  7 14 15  1  8 14  9  7 15  7 11  9  2
> print(calc.r(school.sample, boot.sample, "lsat", "gpa"))
[1] 0.744
>
```

Use the boot function

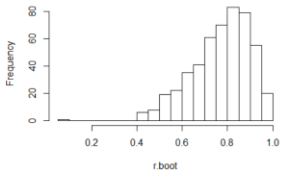
```
> library(boot)
> set.seed(12345)
> R = 500
> boot.out=boot(school.sample, x = "lsat", y = "gpa",
+               R = R, statistic=calc.r)
> print(boot.out)

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = school.sample, statistic = calc.r, R = R, x = "lsat",
    y = "gpa")


Bootstrap Statistics :
    original    bias    std. error
t1*    0.776 -0.005748    0.129298
```


Use the boot.ci function

```
> ci.boot.out=boot.ci(boot.out, type=c("norm","basic", "perc", "bca"))
> print(ci.boot.out)
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 500 bootstrap replicates

CALL :
boot.ci(boot.out = boot.out, type = c("norm", "basic", "perc",
    "bca"))

Intervals :
Level      Normal              Basic
95%   ( 0.5283,  1.0352 )   ( 0.5950,  1.0683 )

Level     Percentile            BCa
95%   ( 0.4837,  0.9570 )   ( 0.4123,  0.9354 )
Calculations and Intervals on Original Scale
Some BCa intervals may be unstable
>
```


"by hand" calculations

```
> r.boot=boot.out$t
> head(r.boot)
          [,1]
[1,] 0.817
[2,] 0.813
[3,] 0.731
[4,] 0.486
[5,] 0.647
[6,] 0.879
>
> boot.mean=mean(r.boot)
> boot.se=sd(r.boot)
> boot.bias=boot.mean-r
>
> print(boot.mean)
[1] 0.770252
> print(boot.se)
[1] 0.129298
> print(boot.bias)
[1] -0.005748
>
```



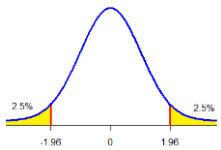
Basic CI

Recall that "usual" CI for a parameter is $\hat{\theta} \pm 1.96 \cdot SE$

This assumes that the standardized empirical distribution of $\hat{\theta}$ approaches the normal distribution

But, actually the CI is $(\hat{\theta} + z_{\alpha/2} \cdot SE,\ \hat{\theta} + z_{1-\alpha/2} \cdot SE)$

And this translates to $(\hat{\theta} + (-1.96) \cdot SE,\ \hat{\theta} + 1.96 \cdot SE)$

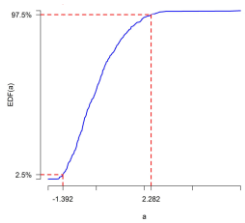


Basic CI

We can generalize the CI as follows:

$CI = (\hat{\theta} + z_{\alpha_1} \cdot SE, \hat{\theta} + z_{\alpha_2} \cdot SE)$

Where z_{α_1} and z_{α_2} are quantiles from the standardized empirical distribution $\hat{F}$ which is estimated by F^*



Basic CI

```
> ci.boot.out$basic[4:5]
[1] 0.595000 1.068333
> z=quantile((r-r.boot)/boot.se, p=c(0.025, 0.975))
> z.low=z[1] # -1.3998
> z.up=z[2]   # 2.2609
> basic.ci=c(r+z.low*boot.se, r+z.up*boot.se)
> print(basic.ci)
    2.5%    97.5%
0.595475 1.066000
>
```

The basic CI length is 0.470

Normal CI

- The normal CI assumes normal distribution
- However, the bias should be considered
- Therefore standardized empirical distribution is assumed to be normal with mean=bias and standard deviation=1
- So the confidence interval is $(\hat{\theta} - bias) \pm z_{1-\alpha/2} \cdot SE$

Normal CI

```
> # Normal CI
> # with correction for bias
> normal.ci=c(r-boot.bias-1.96*boot.se, r-boot.bias+1.96*boot.se)
> print(normal.ci)
[1] 0.528324 1.035172
> ci.boot.out$normal[2:3]
[1] 0.5283286 1.0351674
>
```

The normal CI length is 0.507

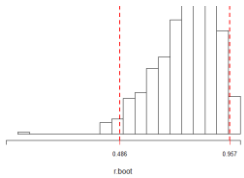
Percentile CI

- Just take the 2.5 and 97.5 percentiles of the empirical distribution

```
> # percentile CI
> percentile.ci=quantile(r.boot, probs=c(0.025, 0.975))
> percentile.ci.length=as.numeric(percentile.ci[2]-percentile.ci[1])
> print(percentile.ci)
    2.5%    97.5%
0.486000 0.956525
>
```

The percentile CI length is 0.471

But, why 2.5% and 97.5%?



Why 2.5% and 97.5% ?

```
> # why 2.5/97.5
> # calculate other possible percentile CI
> # starting from lower.quantile=0.1%
> # up to lower.quantile=4.9%
> all.percentile.ci=data.frame(lower.quantile=0, upper.quantile=0,
+                              lower.ci=0, upper.ci=0, ci.length=0)
> all.percentile.ci=all.percentile.ci[-1,]
> for (j in 1:49){
+   lower.quantile=j/1000
+   all.percentile.ci[j,1]=lower.quantile
+   all.percentile.ci$upper.quantile[j]=1-lower.quantile
+   ci=quantile(r.boot, probs=c(lower.quantile, 1-lower.quantile))
+   all.percentile.ci[j, 3:4]=ci
+   all.percentile.ci$ci.length[j]=ci[2]-ci[1]
+ }
> w=which(all.percentile.ci$ci.length==min(all.percentile.ci$ci.length))
> all.percentile.ci[w,]
   lower.quantile upper.quantile lower.ci upper.ci ci.length
49          0.049          0.951 0.531451    0.942  0.410549
> all.percentile.ci[25,]
   lower.quantile upper.quantile lower.ci upper.ci ci.length
25          0.025          0.975    0.486 0.956525  0.470525
>
```


Bias Correction acceleration (BCa) CI

- Bias correction term: $\hat{z}_0 = \Phi^{-1}\left(F^*(\hat{\theta})\right)$

  The BC term rescales the empirical probability of $\hat{\theta}$ in terms of the standard normal distribution

- Acceleration term: $\hat{a} = \frac{\sum_{j=1}^{R}\left(\bar{\theta}^* - \theta^*_{-j}\right)}{6\left\{\sum_{j=1}^{R}\left(\bar{\theta}^* - \theta^*_{-j}\right)^2\right\}^{1.5}}$

  The acceleration term accounts for situations in which the standard error of an estimator changes depending on the true population value

  Recall that in this example $SE(r) = \sqrt{\frac{1-r^2}{n-2}}$ depends on r

Bias Correction acceleration (BCa) CI

If we sick $1-\alpha$ confidence interval, we set:

$$\alpha_1 = \Phi\left(\hat{z}_0 + \frac{\hat{z}_0 + z_{\alpha/2}}{1 - \hat{a}\left(\hat{z}_0 + z_{\alpha/2}\right)}\right)$$

$$\alpha_2 = \Phi\left(\hat{z}_0 + \frac{\hat{z}_0 + z_{1-\alpha/2}}{1 - \hat{a}\left(\hat{z}_0 + z_{1-\alpha/2}\right)}\right)$$

And then the CI will be the α_1 and α_2 quantiles of the empirical distribution
Note that BCa assumes that n is "large", otherwise the CI will be "unstable"

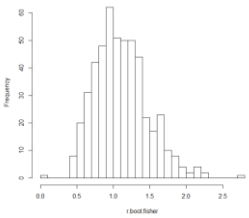
In this example, the length of the BCa CI is 0.523

Transformations

- If we have a distribution that is not "nice", we sometimes apply a transformation to the data
- We can use this trick while bootstrapping:
  1. Take n bootstrap samples
  2. Calculate the statistic from each of the samples
  3. Apply the transformation to the statistics
  4. Calculate bootstrap CI for the transformed statistic
  5. Apply inverse transformation to the CI

Fisher transformation CI

```
> # define transformation and invers transformation
> fisher.z=function(r){
+     z=0.5*log((1+r)/(1-r))
+   return(z)
+ }
>
> inv.fisher.z=function(z){
+   r=(exp(2*z)-1)/(exp(2*z)+1)
+   return(r)
+ }
>
> r.boot.fisher=fisher.z(r.boot)
> hist(r.boot.fisher, breaks=20)
>
```



Fisher transformation CI

```
> # specify transformation and inverse transformationin boot.ci function
> ci.boot.out.fisher.inv=boot.ci(boot.out, type=c("norm","basic", "perc", "bca"),
+                                h=fisher.z, hinv=inv.fisher.z)
> print(ci.boot.out.fisher.inv)
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 500 bootstrap replicates

CALL :
boot.ci(boot.out = boot.out, type = c("norm", "basic", "perc",
    "bca"), h = fisher.z, hinv = inv.fisher.z)

Intervals :
Level      Normal              Basic
95%   ( 0.2404,  0.9319 )   ( 0.1601,  0.9126 )

Level     Percentile            BCa
95%   ( 0.4837,  0.9570 )   ( 0.4123,  0.9354 )
Calculations on Transformed Scale;  Intervals on Original Scale
Some BCa intervals may be unstable
>
```


Summary

	type	lower.cl	upper.cl	ci.length
boot.percentile.optimized		0.5315	0.9420	0.4105
boot.percentile.fisher		0.4837	0.9570	0.4733
boot.basic		0.5950	1.0683	0.4733
boot.percentile		0.4837	0.9570	0.4733
fisher		0.4385	0.9220	0.4835
boot.normal		0.5283	1.0352	0.5068
boot.BCa		0.4123	0.9354	0.5231
boot.BCa.fisher		0.4123	0.9354	0.5231
boot.normal.fisher		0.2404	0.9319	0.6915
boot.basic.fisher		0.1601	0.9126	0.7525
t		0.3987	1.1540	0.7553